Distributing (Fun + Learning): The Distributive Property

Congratulations, you correctly identified a wrong answer! The right answer is: \(-4x+32\)

\(-4(1x-8)\):

4x - 32

\(Y\) \(N\)

Peggy Li
Under the direction of
Professor Susan Rodger
Duke University, July 2011
Note to teachers

NC Standard Course of Study, Grade 6 Mathematics
Competency Goal 5, Objective 5.01:
“Simplify algebraic expressions and verify the results using the basic properties of rational numbers.”  
(Includes using the distributive property)

In addition to the above curriculum standard, this world can also help students practice arithmetic operations (e.g. addition and multiplication) using negative numbers.
Overview

• This tutorial will help you make an Alice world to practice using the *distributive property* and multiply negative numbers.

• You will be shown an expression, and asked to decide whether it correctly uses the distributive property to expand a factored expression.

• Alice programming concepts covered include: variables, parameters, lists/arrays, *if/else* and *while* logic, animations and visual effects, etc.
1. Create a new world using any template. (The tutorial uses Grass, but any other works too.)
2. Click Open, and remember to save your world!
3. Click on Add Objects and go to the People folder. Scroll all the way over to the right, and use hebuilder or shebuilder to build your own character.
4. Even though there are lots of people to choose from, we use Character Builder because it comes with some handy built-in methods.
1. Once you’re done customizing your person, give him or her a name and click **OK**!

2. Your person’s arms will be spread out, like in the picture. Let’s move them down to his or her sides.

   There are two ways to fix the person’s arms. First we’ll take a look at the quicker way.

3. Right click on your character and go to **methods >> set pose >> guide >> UpperBody >> neutral**.

   Note: The character in the named **guide**, but yours might have a different name. Whenever the tutorial refers to **guide**, perform the same action on your own character.
Here’s an alternative that is handy for people that don’t have a neutral pose.

1. Click the + next to guide in the object tree, then the + next to UpperBody and Chest.
2. You should now see a part named rightUpperArm.
3. Right-click on rightUpperArm and select methods. Go to roll >> right. Click on Other and type in 0.22.
4. Repeat Step 1 for the part LeftUpperArm, but this time roll to the left.
1. To add a 3D Text object, click on Add Objects >> Local Gallery and scroll all the way to the right.

2. Click Create 3D Text.

3. Enter “a(bx+c)= ” in the text field.

4. If you want, you can change the font. The tutorial uses Arial bold throughout.

5. Make a second 3D Text object that says “abx+ac ”.

6. Right-click on the first 3D Text in the Object Tree and rename it “expression”. Next, rename the second one “answer”.

3D text objects
1. Let’s change the color of the expression and answer text objects.

2. To do this, click expression in the Object Tree and go to its Properties tab.

3. Click the arrow next to color and choose blue.

4. Repeat steps 2-3 for answer and choose magenta.

Note: You can choose other colors, but the instructions will say blue and magenta. If you use other colors, you might need to create new instructions.
Let’s make two buttons that people will click to answer each question. One will say Y for yes and the other will say N for no.

1. From Local Gallery >> Shapes, add two squares.
2. Rename one yesSquare and the other noSquare.
3. In the Properties tab, change the yesSquare color to green and the noSquare color to red.
4. Right-click on each square and go to methods >> turn to face >> camera.

This method is a more exact than rotating it ourselves. We want the squares as straight as possible.
1. Create two new 3D text objects, one that says “Y” and the other that says “N”. Bold the text.

2. Rename them Yes and No, respectively.

3. Move the Yes so it is right in front of yesSquare. Resize the letter and turn it to face the camera.

   To make the Y as close to yesSquare as possible, push the Y back until it starts going through the square, then pull it forward just a little.

4. Under Yes’s properties tab, set the vehicle of Yes to yesSquare.

5. Repeat Steps 3 and 4 to position No in front of noSquare, and set the vehicle of No to noSquare.
Use the **object control buttons** to move, rotate, and resize the objects until your world looks similar to the picture below:

\[ a(bx+c) = abx + ac \]
1. Click on world in the Object Tree and go to the methods tab. Click create new method and name the method getExpressions.

2. Create a new variable with name a, type number, and value 0 (select other and type in 0).

3. Make the random expressions in the form a(bx+c), where a, b, and c are random integers.
1. Drag a `while` into the method and select `true`.
2. Drag a (from the list of variables at the top) over true, and select `a == 0`.
3. Drag an `a` into the `while` loop. Select `set value`, and choose 1 for now.
4. Under world’s functions tab, find random number and drag it over the 1.

We want nonzero values for a, b, and c so the question isn’t too easy, like 0(2x+0).
By default, the random number generated is a decimal value between 0 and up to but not including 1. The random number includes the minimum but goes up to not including the maximum.

We want integers between and including -10 and 10, so minimum = -10 and maximum = 11. We want integers only so the numbers are easier to work with.

1. Click more at the end of the random number line. Set the minimum to -10, the maximum to 11, and integerOnly to true.
1. Make two copies of the entire `while` loop.

There are two ways to copy something – try both!
- Right click on the line or block of code and select `make copy`.
- Drag a line or block of code to the `clipboard`, then drag the clipboard where you want a copy.

2. In the second loop, change both instances of `a` to `b`.

3. In the third loop, change both instances of `a` to `c`.
1. Create two number variables named ab and ac. Set both values to 0.

2. Drag ab into the bottom of the method and select set value >> expression >> a.

3. Click the arrow by a and choose math >> a* >> expression >> b. This means a x b.

4. Do the same for ac, selecting c instead of b.

5. Under world’s functions tab, find absolute value of a in Math.

6. Drag it over the entire a*b and entire a*c.
1. Create **five** new variables, all type **String** and set to default value. Name them **aString**, **bString**, **cString**, **abString**, and **acString**.

2. Drag **aString** into the method and set its value to default string for now.

3. Under **world’s functions**, find and drag what as a string over default string. Select expressions >> a.

4. Repeat steps 2-3 for the other four variables. Make sure you use the matching number!
1. At the end of world’s functions, find int a as a String.
2. Drag int a as a string over a, b, c, ab, and ac in the five lines you just made. Under expression, reselect each variable name.

What as a string converts a number to a string but makes it a decimal (e.g. 4 becomes 4.0). Using int a as a String removes the decimal (just 4).
1. Create two string variables called `expression` and `rightAnswer` with value default string.
2. Drag an `if/else` into the method and select `true`.
3. Drag the variable `c` over `true` and choose `c <`, then go to `other` and type in `0`.
4. Now drag `expression` into the Do Nothing under the if and set value `>>` default string.
1. Drag **aString** over default string.

2. From world’s **functions**, drag **a joined with b** over **aString**. Go to other and type in (  .

3. Drag in another **a joined with b**, over **just** the left parenthesis (  . Select **expressions** >> **bString**.

4. Drag **a joined with b** over **bString**; go to other and type in **x**.

5. Drag **a joined with b** over **x**. Select **expressions** >> **cString**.

6. Now drag one last **a joined with b** over **cString**, go to **other**, and type in ).
1. Drag the entire **expression set value to** line to the clipboard, then drag the clipboard into the **Do Nothing** under Else to make a copy.

2. Change the **x** to **x+** in the second line.

The next slide will explain why we have the *if/else*. 
UNDERSTANDING WHAT WE JUST DID

• Using many ‘a joined with b’s, we made a string for the expression a(bx+c) with the numeric values of a, b, and c substituted in.

• The \texttt{if/else} was for aesthetic reasons (how it looks). If we have x-2 in parentheses, we want it to say x-2, not x+-2 (note that c = -2).

• If the last number c is < 0, we won’t add + between x and the number. This way, it will just say x-2.

• But \texttt{else} (if c > 0), we do want a plus sign after x. Otherwise, it’d just say x2 instead of x+2!
Setting up the answers

Let’s make an *array* to store four possible answers.

1. Create a new variable called *answers* of type *String*. Check the box by List and change it to *Array*.

2. Add four items with these values:
   - Item 0: $abx+ac$
   - Item 1: $abx-ac$
   - Item 2: $-abx+ac$
   - Item 3: $-abx-ac$
1. First drag **answers** to the bottom of the method.
2. Select **set item <index> to <item> in world.**
   
   `getExpressions.answers >> 0 >> default string.`
3. Drag **a joined with b** over default string and select expressions >> `abString` >> **other > x+.**
4. Drag another **a joined with b** over `x+` and go to expressions >> `acString`.  
5. Make **three** copies of the **set <index> to <item> line** (you should have four duplicate lines).
6. In the second line, change index from 0 to **1** and change `x+` to `x-`. 
1. In the third line, change the index from 0 to 2. Drag a joined with b over abString and select expressions >> abString. Change the first abString to a – sign.

2. In the fourth line, change the index from 0 to 3 and the x+ to x-. Repeat Step 1 to add a – sign before abString.

3. Set the duration of each of the four lines to 0.
Assigning the right answer

- Remember that we started with $a(bx+c)$ and want to expand it to $(ab)x+ac$, i.e. $2(3x-1) = (2*3)x+2*(-1) = 6x-2$.

- $ab$ and $ac$ can be positive or negative, but $|ab|$ and $|ac|$ are both positive (where $|$ $|$ is absolute value). In the example above, $ab = 6$ (positive) and $ac = -2$ (negative), but $|ab| = 6$ and $|ac| = 2$ (both positive).

- Our $answers$ array already contains all four possibilities based on the signs (+ or -), so now we just need to set $rightAnswer$ equal to the item in $answers$ that matches the correct answer.

- The next slide will show the four different possibilities.
Let’s take a look at four possible answer scenarios.

*Note:* We use $ab$ to refer to $|a*b|$.

Scenario 1: $a*b > 0$ and $a*c > 0 \rightarrow +ab$ and $+ac$

\[ +ab, +ac \rightarrow abx + ac \rightarrow \text{index 0 in array} \]

Scenario 2: $a*b > 0$ and $a*c < 0 \rightarrow +ab$ and $-ac$

\[ +ab, -ac \rightarrow abx - ac \rightarrow \text{index 1 in array} \]

Scenario 3: $a*b < 0$ and $a*c > 0 \rightarrow -ab$ and $+ac$

\[ -ab, +ac \rightarrow -abx + ac \rightarrow \text{index 2 in array} \]

Scenario 4: $a*b < 0$ and $a*c < 0 \rightarrow -ab$ and $-ac$

\[ -ab, -ac \rightarrow -abx - ac \rightarrow \text{index 3 in array} \]
1. Drag an if/else into the method and select true.
2. Set the true to $a > 0$, then click the arrow by $a$ and select math $>> a^* >>$ expressions $>> b$.
3. Drag another if/else into the first Do Nothing (under If) and select true.
4. Repeat steps 2 and 3 to change the true to say $a^*c > 0$ (remember to choose $c$, not $b$).

We use $a^*b$ and $a^*c$ instead of $ab$ and $ac$ because $ab$ and $ac$ are always positive ($ab = |a^*b|$ and $ac = |a^*c|$). We want to see if the value of $a^*b$ or $a^*c$ is positive or negative.
Outline of the if/else statements we’ll make. Note that the *else* conditions are implied (opposite of *if*).

If the outer *if* statement $a*b > 0$ is true:
   If the inner *if* statement $a*c > 0$ is true:
       Scenario 1
   If the inner *else* statement $a*c < 0$ is true:
       Scenario 2

If the outer *else* statement $a*b < 0$ is true:
   If the inner *if* statement $a*c > 0$ is true:
       Scenario 3
   If the inner *else* statement $a*c < 0$ is true:
       Scenario 4
1. Drag rightAnswer into the first Do Nothing below if \( a*c > 0 \). Select default string for now.

2. Drag the answers array on top of default string, select ith item from array and then 0.

3. Set the rightAnswer equal to the ith item from array \( \gg 1 \) in the inner else statement. You can repeat steps 1-2 or use the clipboard / make a copy.

4. Drag the entire inner if/else block to the clipboard and back below the outer else to make a copy.

5. Change item 0 to item 2 and item 1 to item 3 in the new if/else.
This is how your nested *if/else* blocks should look:

Go back and make sure the **duration** of every line in this method is set to **0 or false** seconds. If not, change it to 0.
We’re (finally) done with our `getExpressions` method! Now we want to choose a random item from `answers`, that will show up on the screen for the player to see. We want the right answer to appear 50% of the time, but if we choose one of the four options completely randomly, there’s only a 25% chance it’ll come up. Let’s take a look at how we can solve this problem and get a random item from the `answers` array.
We’re going to use a function to return an answer choice.

1. Go to world’s functions tab and create a new function called `getAnswer`. Choose string.
2. Create two new parameters: 1) a string called `rightAnswer` and 2) an array of strings called `allAnswers`.
3. Create a new variable of type number and with name `random`. Create another variable of type string called `answer`.

```plaintext
world.getAnswer (rightAnswer, allAnswers)
random = 1, answer = default string
```
1. Drag **random** into the function and set the value to any number for now.

2. In world’s functions, find **random number**, and drag it over the number you set random equal to.

3. Use the arrow by **random number** to set minimum = 0, maximum = 1, and integerOnly = false.

4. Drag **answer** into the function and set value to expressions >> world.getAnswer.rightAnswer.

5. Drag in an **if/else** and select true, then change the true to **if random >= 0.5**.
1. Drag a **While** loop into the **Do Nothing** under the **If** and select **true**.

2. Drag **answer** onto the true and set **answer == >> expressions >> rightAnswer**.

3. Now drag **answer** into the **While** loop and set its value to **default string** for now.

4. Drag **allAnswers** over **default string**, select **ith item from array**, and choose **0** for now.

5. Change **0** to a **random number with minimum = 0, maximum = 4**, and **integerOnly = true**.

6. Drag **answer** over the **default string** after **Return** on the last line.
We first set `answer` to the `rightAnswer`. Only when the random number is > 0.5 (about 50% of the time), it will be changed to a random item from `allAnswers` until it equals a wrong answer (it is no longer `rightAnswer`).

Your code should now look like this:
1. Create a new world method called play with three string parameters called expression, answer, and rightAnswer.

2. Click on the expression 3D text in your object tree and go to Properties. Drag text into the method and set to expressions >> expression.

3. Repeat Step 2 for the answer text object and the string parameter answer.
1. Drag a **Do Together** into the method.

2. Click on **expression** in the object tree again and go to its **Properties**. Drag **isShowing** into the Do Together and set to **True**. Set **duration** to 0.

3. Repeat step 2 for **answer**.

4. Under **world’s properties**, make a boolean variable called **hasAnswered**, and set it to **false**.

5. Make another world variable of type **string**, name **playerAnswer**, and value “N”.
1. Make a boolean variable called `correct`, set to `false`.
2. Drag an `if/else` statement into the `play` method.
3. Drag `playerAnswer` from world’s Properties over the true and set `playerAnswer==` `>>` expressions `>>` `rightAnswer`.
4. Drag `correct` into the first Do Nothing under `If` and set it to true. Then drag `correct` under `Else` and select `false`.
5. Drag in a `While` and select `true`. Change the condition to `hasAnswered == false`.

We won’t add anything to the `Do Nothing` so that until the player answers (hasAnswered == true), nothing happens.
1. Drag another If/Else into the method. Select true.
2. Drag playerAnswer over true and set playerAnswer == Y.
3. Now drag an If/Else into the Do Nothing under If. Select true, then drag correct over it.
4. Repeat Step 3 or make a copy so you have an If/Else under Else.

Note that playerAnswer is what the user clicks (Y or N) and correct is whether the answer shown is actually right. Once again, we have four possible combinations...
Four different possible answer scenarios:

<table>
<thead>
<tr>
<th>playerAnswer = Y</th>
<th>playerAnswer = N</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>correct = true</strong></td>
<td>✓</td>
</tr>
<tr>
<td>✓ CASE 1</td>
<td>✗ CASE 3</td>
</tr>
<tr>
<td><strong>correct = false</strong></td>
<td>✗</td>
</tr>
<tr>
<td>✗ CASE 2</td>
<td>✓ CASE 4</td>
</tr>
</tbody>
</table>

Note: ✓ means the player either said YES to a right answer or NO to a wrong answer, aka “correctly identified” the answer that appears.
1. Click on guide (or whatever you named your character) in the object tree and go to methods.

2. Drag `guide.yes` under the inner/second `If`.

3. Use `+` in the object tree to get to `guide >> UpperBody >> chest`. Go to methods and drag `say` into the method.

4. Click other and type “Congratulations! You have correctly identified a right answer.” Set duration to 4 seconds.
Scenario 2: playerAnswer = Y and correct = false
(Player thinks answer is right, but it is wrong)

1. Drag `guide.confused` into the inner `Else`.

2. Drag in another `guide.upperBody.chest.say` and type in “Sorry, that was not the right answer. The right answer is: ”

3. From world’s functions, drag `a joined with b` over that string. Click on `expressions` and select `rightAnswer`. Set `duration` to 4 seconds.

4. Drag `Wait` below the `say` and select `2` seconds.
Scenario 3: playerAnswer = N and correct = true
(Player thinks answer is wrong, but it is right)

1. Drag `guide.confused` under the `If` under the outer `Else`.
2. Drag in `guide.UpperBody.chest.say` and type in “I’m sorry, but that was the right answer. Better luck next time!”
3. Set `duration` to 4 seconds.
4. Drag `Wait` below the `say` and select 2 seconds.
Scenario 4: playerAnswer = N and correct = false (Player thinks answer is wrong, and it is wrong)

1. Drag `guide.yes` under both `Else`’s.
2. Drag in `guide.UpperBody.chest.say` and type in “Congratulations, you correctly identified a wrong answer! The right answer is: ”
3. Use `a joined with b` to join the above string with expressions `>>` `rightAnswer`.
4. Set `duration` to 4 seconds.
Your nested if/else statements should look like this:
1. Drag **hasAnswered** from world’s **properties** into the bottom of the play method and set to **false**.

2. Click **expression** in the object tree, go to **properties**, drag **isShowing** below **hasAnswered**, and select **false**.

3. Repeat step 2 for **answer**.

4. Go back and make sure the **duration** of every line in this method is set to **0 or false** seconds.
Creating events

1. Click create new event near the top of the screen; select When the mouse is clicked on something.
2. From object tree, drag yesButton over anything.
3. Drag a Do Together over the Nothing.
4. From world’s properties, drag hasAnswered into Do Together and select true.
5. Drag playerAnswer beneath, and set value to “Y”.
Making new events

1. Use **clipboard** to copy the event you just made, and make three copies.

2. Change **yesButton** to **Yes**, **noButton**, and **No**.

3. For **noButton** and **No**, set **playerAnswer** to “N”.

![Diagram showing the event actions for Yes, noButton, and No]
Connecting the methods

Almost done! Let’s link the methods we made.

1. Go to the `world.getExpressions` method.

2. Drag in the `play` method at the bottom. Choose expressions >> `expression` >> expressions >> `default string` >> expressions >> `rightAnswer`

3. Drag the `getAnswer` function over `default string`, and drag variables `rightAnswer` and `answers` over `rightAnswer` and `answerArray`.

```
world.play  expression = expression  answer =
world.getAnswer  rightAnswer = rightAnswer  answerArray = answers  rightAnswer = rightAnswer
```
1. Make sure you’ve downloaded instructions.png from the website and saved it somewhere.

2. Go to File >> Make Billboard and import it.

3. Rotate, resize, and reposition the billboard so it covers most/all of the screen.

4. From instruction’s properties, drag isShowing over world.my first method on the event line When the world starts, do and select true.

5. Make sure duration is set to 0.
1. Create a new event, **when a key is typed**.
2. Change **any key** to **space** from the dropdown.
3. From world’s methods tab, drag **world.my first method** over **Nothing**.

Remember that when the world starts, we first display the instructions. When the user presses Space, **my first method** is called, and we want to set the instructions to invisible before the game itself begins – we’ll do this on the next slide.
1. In world.my first method, drag in instruction’s isShowing at the top. Set to false and duration = 0.

2. Drag in a loop, then choose the number of questions you want the world to ask. The tutorial shows 10, but you can ask more or less.

3. Drag world.getExpressions into the Do Nothing.

4. Below that, have the guide say “Congratulations, you have answered all the questions. That’s all for now!” with duration = 4 seconds.
Your code for *my first method* should look like this:

1. Make sure that `isShowing` is set to false for the expression and answer 3D text (in properties).
Reaching THE END

Congratulations! You're done with the tutorial! Now click Play to test it out.

-9(10x-7)

-90x+63

Y  N